1 Graph theoretical approaches to delineate dynamics of biological processes

TERESA M. PRZYTYCKA and ELENA ZOTENKO[†]

National Center for Biotechnology Information, National Library of Medicine,
National Institutes of Health, Bethesda, MD 20894,USA

1.1 INTRODUCTION

Graphs are used in Computational Biology to model the relationships between biological entities. For example, experimentally determined protein interactions are commonly represented by a graph, the so-called *protein interaction network*, where proteins are nodes and every pair of interacting proteins is connected by an edge. Even though such a representation may not capture all the complexity of protein interactions in underlying biological processes, the study of the topological properties of these networks has become an important tool in searching for general principles that govern the organization of molecular networks. For example, it was observed that in protein interaction networks some types of small-size subnetworks are much more abundant than would be expected by chance [53]. The discovery of these overrepresented subnetworks or *network motifs* has led to investigation of their information processing properties [64] and network evolution mechanisms that could account for their emergence [52]. Usage of graph theoretical tools is not limited to the study of protein interaction networks, graphs are also used to model metabolic networks (processes), gene co-expression, gene co-regulation, phylogenies, etc.

In general, graphs are not required to have any type of regularity. This makes them very flexible combinatorial objects, which are able to represent complex and diverse relationships. In practice, however, graphs that model real world phenomena often belong to families of graphs with a special structure, which can be exploited to gain

[†]this author is also affiliated with Department of Computer Science, University of Maryland,College Park, MD 20742, USA

an insight into the phenomenon that generated the graph. To clarify this statement we start with a following toy example taken from everyday life.

Example: Elena decided to walk 40 miles to raise funds for an important cause. Her friends provide her with support by walking along her, but each of them walks only for 10 miles (see Figure 1.1(a)). Her husband, Julian, volunteers to document the event and takes a group picture every time a new supporter joins Elena (see Figure 1.1(b)). After the event is completed Julian handles Elena a box with photographs. Elena notices that the pictures are not ordered and then she learns that Julian lost somewhere the film. Can she reconstruct the order her supporters joined the walk without the film, i.e., can she use the information in Figure 1.1(b) to tell that her friends joined the walk in the following order (Merrick, Nilani, Dami, Teresa, Raja, Praveen)? If Julian had lost the film before developing it (so Elena does not have her precious pictures) but her supporters remember their walking partners, however they do not remember the order in which these partners joined, would she still be able to reconstruct the history of events? Interestingly, if except for the very beginning and very end she never walked alone and remembers a person who supported her first she can reconstruct this order: in the first case she would be able to recover the order completely; in the second case she still would be able to reconstruct the order except for the relative placement of Dami and Teresa; she would not be able to tell whether Dami joined the walk before Teresa or the other way around.

In the example above, Elena exploits the special structure of the supporters overlap graph in Figure 1.1(c) to understand the "real world phenomenon", the participation of her friends in the fund raising event in Figure 1.1(a). The graph in Figure 1.1(c) is an *interval graph*, meaning that there is a set of intervals on a real line such that vertices of the graph are in one-to-one correspondence with the intervals in the set and there is an edge between a pair of vertices if and only if the corresponding intervals intersect; the set of intervals is called an *interval representation* of the graph. Interval graphs are a special case of *intersection graphs*, graphs whose vertices are in one to one correspondence with a family of sets such that there is an edge between a pair of vertices if and only if the corresponding pair of sets have a non-empty intersection. Coming back to our example, the supporters overlap graph in Figure 1.1(c) is an interval graph with one possible interval representation shown in Figure 1.1(a). Given the graph in Figure 1.1(c) Elena won't be able to reconstruct the history of events up to the smallest detail, such as Merrick joined the walk 8 miles before Nilani, but she would be able to tell that all possible valid (Merrick is the first to join the walk and everybody walks for exactly 10 miles) interval representations of this graph result in the same order (up to relative placement of Dami and Teresa) of her friends joining the walk.

In this chapter we will demonstrate how graph theoretical tools are used in Computational Biology to elucidate the dynamics of biological processes. In particular, we will show applications of the well studied graph family known as *chordal graphs*. Chordal graphs are exactly these graphs that are intersection graphs of subtrees of a tree, and therefore they include interval graphs which can be seen as intersection graphs of subtrees of a path (a degenerate tree). We start with a background infor-

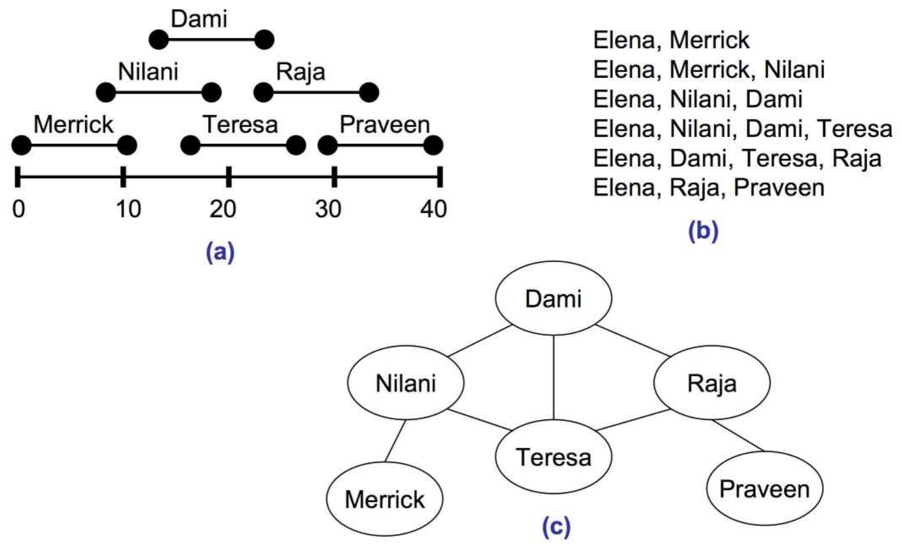


Fig. 1.1 Elena's story. **(a)** The order in which Elena's friends, Merrick, Nilani, Dami, Teresa, Raja, and Praveen, join the walk. Each friend is represented by an interval showing his/her stretch of the walk. **(b)** Julian's pictures. There are six pictures showing the participants when each friend joins the walk. **(c)** The supporters overlap graph: Elena's friends are nodes and there is an edge between two friends if they were walking together.

mation on graph theoretical tools used to deal with chordal graphs (see Section 1.2). We then proceed to show how these tools are applied to two problems in Computational Biology: phylogenetic tree reconstruction (see Section 1.3) and formation of multi-protein complexes (see Section 1.4). In both applications, structure of a certain graph is exploited (in a manner similar to the toy example above) to elucidate the dynamic behavior of the underlying biological process. In the first application we are interested in the dynamics of evolution, i.e., the order in which the taxa evolved from a common ancestor. In the second application we are interested in the dynamics of multi-protein complex formation during a biological process, such as cell signalling, i.e., how multi-protein complexes are formed during the process and the order in which proteins join these complexes.

1.2 GRAPH THEORY BACKGROUND

The purpose of this section is to provide the reader with an overview of relevant graph theoretic results for chordal, interval, and cograph graph families. We state here results that are used in the biological applications of these graph families discussed in latter sections. For a thorough treatment of chordal graphs and interval graphs we refer the reader to now a classical book by Golumbic [35]; other excellent references are a recent book on intersection graph theory by McKee and McMorris [50], a chapter "An introduction to chordal graphs and clique trees" by Blair and Peyton in [33], and a set of lecture notes by Shamir [63]. For an overview of structural and algorithmic properties of cographs we refer the reader to the paper by Corneil *et al.* [19]; modular decomposition is surveyed in a paper by Mohring and Radermacher[54], a nice overview can be also found in a chapter "Decompositions and forcing relations in graphs and other combinatorial structures" by McConnel in [36].

We assume that all graphs are undirected and connected. We denote by $G = (V, E)$ a graph with a set of vertices V and a set of edges E. Given a graph $G = (V, E)$, a subgraph $G' = (V', E')$ is an *induced subgraph* of G if V' is a subset of V and E' contains all the edges of the original graph whose both endpoints are in V'; we may also say that G' is a subgraph of G *induced* by V'. For a vertex $v \in V$, we use $\mathcal{N}(v)$ to denote the set of v's neighbors in G, i.e., $\mathcal{N}(v) = \{u \mid (v, u) \in E\}$. We use "$-$" to denote set difference operation such that for two sets X and Y the set $X - Y$ contains elements that are in X but not in Y.

1.2.1 Chordal Graphs

In a cycle a *chord* is an edge that connects two non-consecutive vertices of the cycle. For example, a cycle $\{a, b, c, d\}$ in Figure 1.2(a) has a chord (b, d). A *chordal graph* is a graph that does not contain chordless cycles of length greater than three; other names given to graphs having this property are *rigid circuit graphs* and *triangulated graphs*. Chordality is a *hereditary graph property*, meaning that any induced subgraph of a chordal graph is chordal.

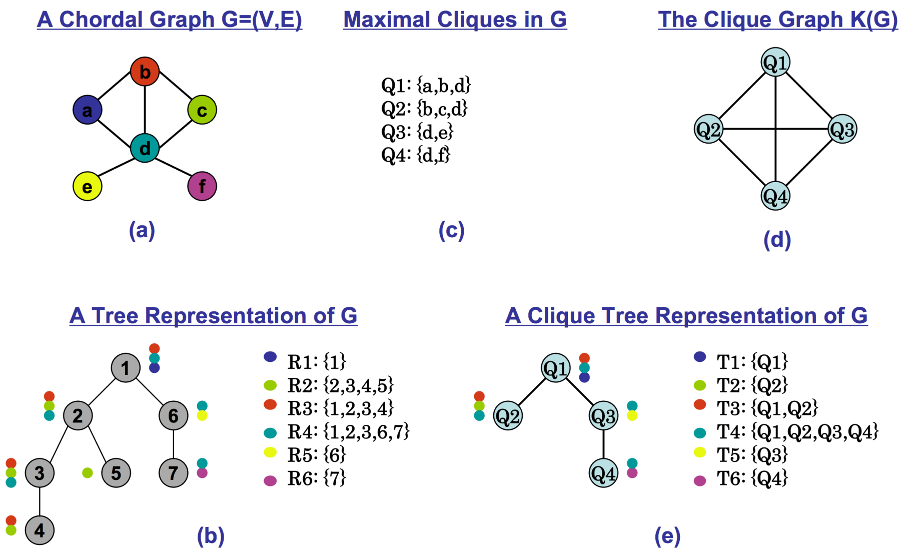


Fig. 1.2 **(a)** A chordal graph $G = (V, E)$. **(b)** A tree representation of G: the tree is on the left and the family of subtrees is on the right. Every subtree ,R_i, is schematically shown by putting a colored circle next to its nodes on the tree. For example, R_3 is shown by red circles. **(c)** The set of maximal cliques in G. There are four maximal cliques in the graph, Q_1, Q_2, Q_3, and Q_4. **(d)** The clique graph of G. The clique graph is the intersection graph of $\{Q_1, Q_2, Q_3, Q_4\}$. **(e)** A clique tree representation of G: the clique tree is on the left and the family of subtrees is on the right. It should be noted that a clique tree is a valid tree representation of a chordal graph. Indeed, every vertex in the graph corresponds to a subtree of the clique tree and two vertices are adjacent if and only if their corresponding subtrees intersect.

In a graph an ordering of vertices $\{v_1, ..., v_n\}$ is a *perfect elimination ordering* (PEO) if and only if for every position i the subgraph induced by the neighbors of v_i that appear later on in the ordering is complete, i.e., the subgraph induced by $\mathcal{N}(v_i) \cap \{v_{i+1}, ..., v_n\}$ is complete. For example in the graph of Figure 1.2(a), the ordering $\{a, b, c, e, f, d\}$ is a PEO while the ordering $\{a, b, c, d, e, f\}$ is not. It was shown by Fulkerson and Gross [26] that only chordal graphs can have a PEO.

Theorem 1 (Fulkerson&Gross1965) *A graph is chordal if and only if there exists a perfect elimination ordering of its vertices.*

This alternative characterization of chordal graphs is used by two linear time chordal graph recognition algorithms [59, 66]. Given a graph, both algorithms produce an ordering of its vertices which is a PEO if and only if the input graph is chordal. Therefore, to determine whether the input graph is chordal it suffices to check that the ordering output by the algorithm is a PEO. The earliest algorithm, due to Rose and Tarjan [59], uses a *Lexicographic Breadth-First Search*(LexBFS), a modified version of the widely known *Breadth First Search* [17] algorithm, to order the vertices of the graph.

A *maximal clique* in a graph is a subset of vertices that form a maximal complete subgraph. Given a graph G, we will use $\mathcal{Q}(G)$ to denote the set of all maximal cliques in G and $K(G)$ to denote the *clique graph* of G, where vertices of $K(G)$ are maximal cliques in G and there is an edge between a pair of vertices (maximal cliques) if their intersection is not empty. As an illustration consider the graph in Figure 1.2(a). This graph has four maximal cliques, which are shown in Figure 1.2(c). The clique graph $K(G)$ is shown in Figure 1.2(d); it has four vertices Q_1, Q_2, Q_3, and Q_4, and is complete as every pair of vertices (maximal cliques) has a non-empty intersection. (In this case all maximal cliques contain vertex $d \in V$ of the original graph G.)

Even though computing all maximal cliques of a general graph is a difficult problem [28], all maximal cliques of a chordal graph can be computed efficiently. Moreover, the number of maximal cliques in a chordal graph is at most $|V|$. (For details please refer to Section 4.2.1 in the chapter by Blair and Peyton [33].)

Let $\mathcal{F} = \{R_1, ..., R_n\}$ be a family of subsets. The *intersection graph of* $\mathcal{F}$ is a graph $G = (V, E)$ where $V = \mathcal{F}$ and $E = \{(R_i, R_j) \,|\, R_i \cap R_j \neq \emptyset\}$, i.e., the vertices of the graph are the subsets in $\mathcal{F}$ and there is an edge between two vertices (subsets) if their intersection is not empty. It can be shown that every graph is isomorphic to the intersection graph of some family of subsets; the family of subsets can be thought as an alternative representation of the graph and is called an *intersection representation* of the graph. A variety of well known graph classes can be characterized by putting restrictions on intersection representations of graphs in the class. For example, an *interval graph* is isomorphic to the intersection graph of a family of closed intervals on the real line and a *chordal graph* is isomorphic to the intersection graph of a family of subtrees of a tree.

Even though the study of chordal graphs goes back to 1958, the characterization in terms of allowable intersection representations was given only in seventies [68, 31, 13]. In particular, it was established that a graph is chordal if and only if it is isomorphic to the intersection graph of a family of subtrees of a tree; the tree and the family of subtrees are called a *tree representation* of the chordal graph. Figure 1.2(b) shows a tree representation of a chordal graph in Figure 1.2(a). Moreover, it was shown that every chordal graph $G = (V, E)$ has a special tree representation, the so-called *clique tree representation*, in which the tree is a spanning tree of $K(G)$ and the family of subtrees $\mathcal{F} = \{T_v \mid v \in V\}$ is defined by setting each T_v to the set of maximal cliques that contain v. For example, Figure 1.2(e) shows a clique tree representation for a chordal graph in Figure 1.2(a). This is summarized in the following theorem:

Theorem 2 (Walter1972, Gavril1974, Buneman1974) *Let $G = (V, E)$ be a graph. The following statements are equivalent:*

1. *G is a chordal graph.*

2. *G is isomorphic to the intersection graph of a family of subtrees of a tree.*

3. *There exists a spanning tree of the clique graph $K(G)$ such that for every $v \in V$ the subgraph of this tree induced by the set of maximal cliques containing v, $\{Q \mid Q \in \mathcal{Q}(G)$ and $v \in Q\}$, is connected.*

Given a chordal graph, all possible clique tree representations can be efficiently computed. One approach [7] is based on the fact that clique trees are exactly maximum weight spanning trees of the clique graph $K(G)$, where the weight function on the edges of $K(G)$ is defined as the amount of overlap between two maximal cliques, i.e., $w(Q', Q'') = |Q' \cap Q''|$. Thus, in order to compute all possible clique tree representations of a chordal graph, one simply needs to compute all maximum weight spanning trees of the clique graph $K(G)$, for example by using an algorithm from [32]. Another approach [39] builds on a connection between the edges of a clique tree of a chordal graph and the set of minimal vertex separators in the graph.

Given a graph $G = (V, E)$ not necessarily chordal, one is often interested in finding a set of edges E' such that addition of E' to the graph makes it chordal; the set of edges that does the job is called a *triangulation* of G. As a complete graph is chordal by definition, any graph can be trivially triangulated by setting E' to be the set of all non-edges in the graph, $E' = (V \times V) - E$. One may further ask for a triangulation that possesses additional properties. A *minimal triangulation* of a graph is a triangulation that is not properly contained in any other triangulation. A minimal triangulation can be found efficiently [59] using a variant of LexBFS algorithm for recognition of chordal graphs. A *minimum triangulation* of a graph is the triangulation with the smallest number of edges. Even though finding a minimum triangulation of a graph is a difficult problem [71], there are *fixed-parameter tractable* solutions [14, 45]. For example, an algorithm in [14] takes $O(\frac{4^k}{(k+1)^{3/2}}(|V| + |E|))$ to find a minimum triangulation of $G = (V, E)$ when G has a triangulation whose size does not exceed k. Therefore, if the size of minimum triangulation is small it can be found efficiently.

1.2.2 Interval Graphs

An *interval graph* is any graph that is isomorphic to the intersection graph of a family of intervals on a real line; the family of intervals is called *interval representation* or sometimes an *interval realizer* of the graph. Not every graph has an interval representation; consider for example a chordless cycle of length four. The "invention" of interval graphs is commonly attributed to the Hungarian mathematician Gyorgy Hajos who in 1957 posed the problem of characterizing this family of graphs. Interval graphs also appear in the work of the American biologists Seymour Benzer [6] who used them to support his hypothesis that genetic material is organized into a structure having linear topology.

The first linear time algorithm for recognizing interval graphs is due to Booth and Leuker [11]. In their paper the authors show how to test whether a family of subsets of some ground set U has a *consecutive ones* property, meaning that the members of the family can be linearly ordered in a way such that for every element in U the subsets containing it are consecutive in the linear order. Therefore, according to the theorem below, an interval graph is recognized by testing whether the set of its maximal cliques has a consecutive ones property.

Theorem 3 (Gilmore&Hoffman1964) *A graph is an interval graph if and only if its maximal cliques can be ordered in a linear fashion such that for every vertex in the graph the set of maximal cliques that contain it is consecutive.*

The above characterization implies that interval graphs are chordal. Indeed, if maximal cliques of a chordal graph can be arranged in a tree then maximal cliques of an interval graph can be arranged on a path. Therefore, interval graphs are exactly these chordal graphs that have a clique tree representation which is a path.

In a graph $G = (V, E)$ an ordering of vertices $\{v_1, ..., v_n\}$ is an *interval ordering* (I-ordering) if and only if for every pair of positions $i < j$ the following holds: if $(v_i, v_j) \in E$ then $(v_i, v_k) \in E$ for every $i < k < j$. Recently another linear time algorithm for recognition of interval graphs was proposed [18], which utilizes the fact that only interval graphs can have an I-ordering. The main idea is to use a multi-sweep LexBFS algorithm to produce an ordering of the vertices of a graph, which is an I-ordering if and only if the input graph is an interval graph.

1.2.3 Modular Decomposition and Cographs

A *module* in a graph $G = (V, E)$ is a set of vertices, X, that have exactly the same set of neighbors in $V - X$, i.e., for every pair of vertices u and v in X the following holds $\mathcal{N}(u) \cap (V - X) = \mathcal{N}(v) \cap (V - X)$. For any vertex v, the set $\{v\}$ trivially satisfies the requirement for being a module and so does the set of all vertices in the graph, V; these sets are called *trivial modules*.

A graph that only has trivial modules is *prime*; for example, the graph in Figure 1.3(a) is prime, while the graph in Figure 1.3(b) is not. A non-prime graph will have other modules in addition to the trivial modules. Two modules in a graph *overlap* if they share vertices but neither module properly contains the other. A module

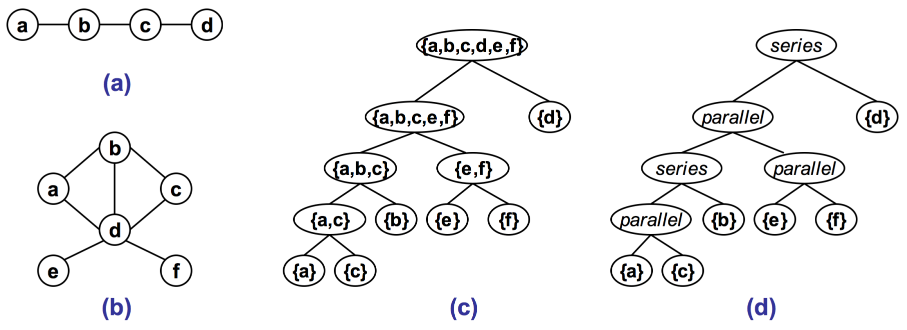


Fig. 1.3 **(a)** A prime graph. **(b)** A non-prime graph. **(c)** The modular decomposition tree of the graph in **(a)**. **(d)** The modular decomposition tree can be used to derive a Boolean expression for the maximal cliques in a graph. The corresponding modular decomposition tree. The Boolean expression is constructed by moving along the tree from the leaves to the root, replacing each "series" node with an $\wedge$ operator and each "parallel" node with an $\vee$ operator. The Boolean expression for the cograph in **(b)** is $(((a \vee c) \wedge b) \vee e \vee f) \wedge d$.

is *strong* if it does not overlap any other module in the graph and *weak* otherwise; by definition trivial modules are strong modules.

The strong modules in a graph $G = (V, E)$ can be organized into a hierarchical structure where every module is attached to the smallest module that contains it. It can be argued that this construction results in a unique tree, the *modular decomposition tree* of the graph, with the trivial modules of the form $\{v\}$ being the leaves of the tree, the module V being the root, and all other strong modules being the internal nodes. The modular decomposition tree of the graph in Figure 1.3(b) is shown in Figure 1.3(c). This graph has eleven modules, all of which are strong.

Even though weak modules of a graph do not directly appear in the modular decomposition tree, it can be shown that every weak module is a union of strong modules that are directly attached to the same internal node in the modular decomposition tree. When this happens the internal node is labeled as *degenerate*; internal nodes that are not degenerate are labeled as *prime*. Furthermore, the union of any subset of children of a degenerate node is a module (necessarily weak). Therefore, the modular decomposition tree captures all modules in the graph: the strong modules are the nodes of the tree and the weak modules are the unions of children of degenerate internal nodes.

Let X be a module in a graph $G = (V, E)$ represented by internal node of the modular decomposition tree and let $\mathcal{C}$ be the set of modules that correspond to its children. A *quotient graph* associated with X is obtained by contracting every module in $\mathcal{C}$ into one node in the subgraph of G induced by X, G_X. For any pair of modules Y and Y' in $\mathcal{C}$, either all edges $Y \times Y'$ belong to E or none does ($(Y \times Y') \cap E = \emptyset$). Therefore, the quotient graph associated with X completely specifies the edges of G_X that are not within one module in $\mathcal{C}$. Moreover, it can be shown that the quotient graph associated with a module that corresponds to a degenerate node is either a complete graph or a complement of a complete graph. If we label degenerate nodes as *series* whenever the corresponding quotient graph is complete and *parallel* otherwise, and record the structure of quotient graphs associated with prime nodes, then the modular decomposition tree together with this additional information completely specifies the structure of the graph.

A *complement reducible graph* (a cograph) can be recursively defined in the following manner: (i) a single vertex graph is a cograph; (ii) if G_1, ..., G_k, are cographs then so is their union $G_1 \cup G_2 ... \cup G_k$; (iii) if G is a cograph then so is its complement $\bar{G}$; A pair of nodes, u and v, in a graph are *siblings* if they have exactly the same set of neighbors, i.e., $\mathcal{N}(u) - \{v\} = \mathcal{N}(v) - \{u\}$. If the nodes of the pair are connected by an edge, we call them *strong siblings* and *weak siblings* otherwise. The following theorem summarizes some of the structural properties of cographs given in the paper by Corneil *et al.* [19].

Theorem 4 *Let $G = (V, E)$ be a graph. The following statements are equivalent.*

- *G is a cograph.*
- *Every non-trivial induced subgraph of G has a pair of siblings.*
- *G does not contain an induced subgraph isomorphic to a path of length four (P_4).*

Cographs are exactly graphs with the modular decomposition tree without prime modules. Therefore the modular decomposition tree of a cograph with the "series"/"parallel" labeling of nodes provides an alternative representation of the graph. This representation is closely related to the *cotree* representation for cographs [19]. In particular, the modular decomposition tree can be used to generate a Boolean expression describing all the maximal cliques in a cograph and obtain efficient algorithms for other otherwise difficult combinatorial problems [19]. The Boolean expression is constructed by moving along the tree from the leaves to the root, replacing each "series" node with an $\wedge$ operator and every "parallel" node with an $\vee$ operator. For example, Figure 1.3(d) shows how to obtain the Boolean expression for the graph in Figure 1.3(b). For a cograph the modular decomposition tree can be constructed in linear time [20].

1.3 RECONSTRUCTING PHYLOGENIES

Consider a set of taxa, where each taxon is represented by a vector of attributes, the so-called *characters*. We assume that every character can take one of a finite number of states and the set of taxa evolved from a common ancestor through changes of states of the corresponding characters. For example, the set of taxa can be described by columns in multiple sequence alignment of protein sequences. In this case each column in the alignment is a character that can assume one of twenty possible states.

Parsimony methods seek a phylogenetic tree that explains the observed characters with the minimum number of character changes along the branches of the tree.

In our working example for this section the set of taxa includes eight species shown in Figure 1.4 (a); each species is described by two binary characters. As there are $\frac{(2n-5)!}{2^{n-3}(n-3)!}$ unrooted binary trees on n labeled vertices [15], there are $\frac{11!}{2^5 5!} = 10,395$ possible phylogenetic trees for the set of species in our example. One such tree is shown in Figure 1.4(b). Once the tree topology is fixed, an optimal assignment/assignments of the character states to the internal nodes can be efficiently computed [25]; the assignment of characters in Figure1.4(b) is optimal for this tree topology and requires three character changes.

We call a phylogenetic tree *perfect phylogeny* if every character state arose only once during evolution or in other words the subgraph of the tree induced by the nodes having this character state is connected. The phylogenetic tree in Figure 1.4(b) is not a perfect phylogeny as the character state 0 for the character "intron 256" arose twice, once in the part of the tree defined by **Sc** and **Sp** and another time in the part of the tree defined by **Dm** and **Ag**. Given a phylogenetic tree, the number of changes due to a specific character is bounded from below by the number of states this character assumes minus one. It is easy to see that the lower bound is achieved only when each character state induces a connected subgraph of the tree; in the phylogenetic tree of Figure 1.4 (b) the character "intron 105" achieves the lower bound, while the character "intron 256" does not. Therefore, a perfect phylogeny is the best tree in a sense that it achieves this lower bound for every character. Perfect phylogeny often does not exist and we start this section with an example of how Chordal Graph Theory can be used to address the `Character Compatibility Problem`: Given a set of taxa, does there exist the perfect phylogeny for the set?

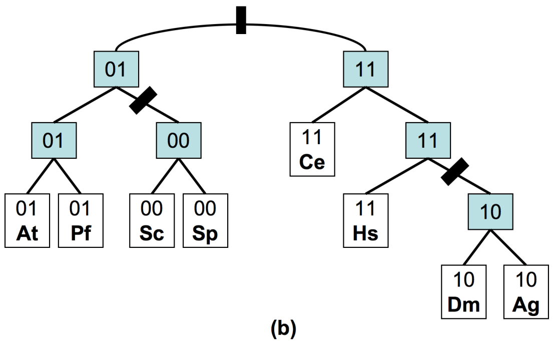
When a set of taxa admits perfect phylogeny we say that the characters describing the set are *fully compatible* or just *compatible*. The compatibility criteria is quite restrictive, in the case of intron data, for example, it means that for every intron the transition from "0" state to "1" state occurred only once during evolution. We conclude this section by showing how Chordal Graph Theory can be used to relax the compatibility criteria in a meaningful way when taxa are described by a set of binary characters.

1.3.1 Perfect Phylogeny and Triangulating Vertex-Colored Graphs

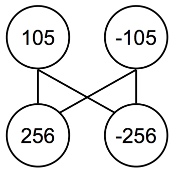
From the set of input taxa we can construct a *partition intersection graph* in the following manner: (i) introduce a vertex for every character state; (ii) put an edge between two vertices if the corresponding character states are observed in one or more taxa together. In our working example the partition intersection graph will have four vertices, 105 (state "1" of the character "intron 105"), -105 (state "0" of the character "intron 105"), 256 (state "1" of the character "intron 256"), and -256 (state "0" of the character "intron 256") (see Figure 1.4 (c)). The name "partition intersection graph" is due to the fact that each character state corresponds to a subset

	105	256
Ag	1	0
At	0	1
Ce	1	1
Dm	1	0
Hs	1	1
Pf	0	1
Sc	0	0
Sp	0	0

(a)



(b)



(c)

Fig. 1.4 A set of eight species: *Anopheles gambiae* (**Ag**), *Arabidopsis thaliana* (**At**), *Caenorhabditis elegans* (**Ce**), *Drosophila melanogaster* (**Dm**), *Homo sapiens* (**Hm**), *Plasmodium falciparum* (**Pf**), *Saccharomyces cerevisiae* (**Ag**), and *Saccharomyces pombe* (**Sp**). **(a)** The species are described by binary characters which correspond to the presence (value of 1) or absence (value of 0) of introns. This is truncated data limited to just two introns (105 and 256) out of about $7,236$ from the study of Rogozin *et al.* [58]. **(b)** A phylogenetic tree: the leaves are the species in the set and are labeled with the input character states; the internal nodes are ancestral species and are labeled with the inferred character states. This particular tree requires three character changes , which are marked with solid bars on the corresponding edges.**(c)** The character overlap graph. There are four vertices, one vertex per character state, 105 (state “1” of the character “intron 105”), -105 (state “0” of the character “intron 105”), 256 (state “1” of the character “intron 256”), and -256 (state “0” of the character “intron 256”). Two vertices are connected by an edge if corresponding character states are observed together in some taxon. The edge $(105, -256)$, for example, is due to species **Ag** and **Dm**.

of taxa, the taxa that have this character state, and the subsets of character states of a character partition the set of taxa under consideration.

There is an important connection between partition intersection graphs and the `Character Compatibility Problem`. Indeed, if a set of taxa admits perfect phylogeny then there exists a phylogenetic tree, where for each character state the tree vertices having this state form a subtree. As there is an edge in the partition intersection graph between every pair of character states whose subtrees intersect in the leaves of the phylogenetic tree, this graph is either chordal or can be triangulated without introducing edges between vertices that correspond to the states of the same character. (Additional edges may be necessary to account for subtree intersection which occurs only at internal nodes of the phylogenetic tree.) The partition intersection graphs were used by Buneman [13] (in his paper the author refers to these graphs as *attribute overlap graphs*) to show that the `Character Compatibility Problem` reduces in polynomial time to the `Triangulating Vertex Colored Graph Problem`. In the latter problem we are given a graph $G(V, E)$ and a proper coloring of its vertices, $c : V \rightarrow Z$. A vertex coloring is proper if there does not exist an edge in G whose endpoints are assigned the same color by the coloring. We want to determine if there exists a chordal graph $\hat{G}(V, \hat{E})$ such that $E \subset \hat{E}$ and $\hat{G}$ is properly colored by c, i.e., no edges between vertices of the same color were introduced in the process of triangulating G. If such chordal graph exists we say that G can be c-triangulated.

Theorem 5 (Buneman1974) *A set of taxa has a perfect phylogeny if and only if the corresponding partition intersection graph can be c-triangulated, where vertex coloring function c assigns the same color to the character states of the same character and different colors to the character states of different characters.*

Kannan and Warnow [43] showed the polynomial time reduction in the opposite direction: from the `Triangulating Vertex Colored Graph Problem` to the `Character Compatibility Problem`, thus establishing that the two problems are equivalent. This result was later used by Bodlaender *et al.* [9] to show that the `Character Compatibility Problem` is NP-complete. Even though the `Character Compatibility Problem` is hard in general, there are efficient algorithms when one or more of the problem's natural parameters are fixed: n the number of taxonomic units, k the number of characters, and r the maximum number of states per character. Later on we will see how to apply the Buneman's theorem to derive a polynomial time solution for two characters $k = 2$. For three characters there is a series of algorithms that run in linear time [43, 41, 10]. For arbitrary fixed k there is an $O(r^{k+1}k^{k+1} + nk^2)$ algorithm due to McMorris *et al.* [51]. When the number of character states is bounded the problem can also be solved efficiently. There is a simple linear time algorithm to test if any number of binary characters is compatible due to Gusfield [38]. For four-state characters there is an $O(n^2k)$ algorithm due to Kannan and Warnow [44]. For arbitrary fixed r there is an $O(2^{3r}(nk^3 + k^4))$ algorithm due to Agarwala and Fernandez-Baca [2].

The Buneman's theorem can be used to readily derive a well known test for checking whether a pair of binary characters is compatible. The test is attributed to E.O.Wilson [69]; it says that a pair of binary characters is compatible if and only if there does not exist a set of four taxa having all possible character states, 00, 01, 10, and 11. The same test can be derived through application of the Buneman's theorem. According to the theorem a pair of binary characters is compatible if and only if the corresponding partition intersection can be c-triangulated. As there are only two binary characters the partition intersection graph is bipartite and each set of the bipartition contains two vertices (see for example Figure 1.4 (c)). Such a graph is either acyclic and therefore can be trivially c-triangulated, or it contains a square and therefore does not have a c-triangulation as any attempt to eliminate the square would add an edge between two vertices of the same color. The square in the partition intersection graph corresponds to the presence of the four taxa with all possible combinations of character values: 00, 01, 10, and 11, where 00, for example, means that both characters have state "0". The compatibility test can be extended to a pair of characters with more than two states ($r > 2$). In this case the partition intersection graph would still be bipartite and the number of vertices in each bipartition is r. It can be easily shown that this graph can be c-triangulated if and only if it is acyclic. Therefore testing compatibility of two characters reduces to testing whether the partition intersection graph is acyclic which can be done efficiently, for example using any of the graph search algorithms such as BFS or DFS [17].

1.3.2 Character Stability

Assume that we are dealing with a set of characters which are difficult to gain but relatively easy to lose. A classic example of such characters are introns [23]. Introns are non-coding DNA sequences that interrupt the flow of a gene coding sequences in eukaryotic genes. They are remarkably conserved between some lineages (e.g. between Arabidopsis and Human), but they are lost at a significant rate in other organisms (e.g. Worm) [58]. Parsimony methods applied to introns produced an incorrect tree [58] indicating that the data contains misleading characters. One way of eliminating such misleading characters is to restrict attention to a maximum set of compatible characters. However, under the condition that the characters are hard to gain but are frequently lost, a large enough set of compatible characters may not exist. To address this problem Przytycka [56] proposed a new consistency criterion called *stability criterion*.

The definition of the stability criterion is phrased as a property of a graph closely related to the partition intersection graph and called a *character overlap graph*. A character overlap graph for a set of taxa is a graph $G = (V, E)$, where V is a set of characters, and $(u, v) \in E$ if there exists a taxon T in the set such that both u and v are present in T. Note that the character overlap graph is simply a subgraph of the partition intersection graph for a set of binary characters that is induced by the set of characters in state "1".

To motivate the concept of stability, consider a set of characters A, B, C, D and a set of four taxa described respectively by character pairs: (A, B), (B, C), (C, D), and (D, A). That is the first taxon has characters A and B in state "1" (and the rest in state "0"), second B and C in state "1", and so on. In such a case the corresponding character overlap graph is simply a square (see Figure 1.5(a)). There

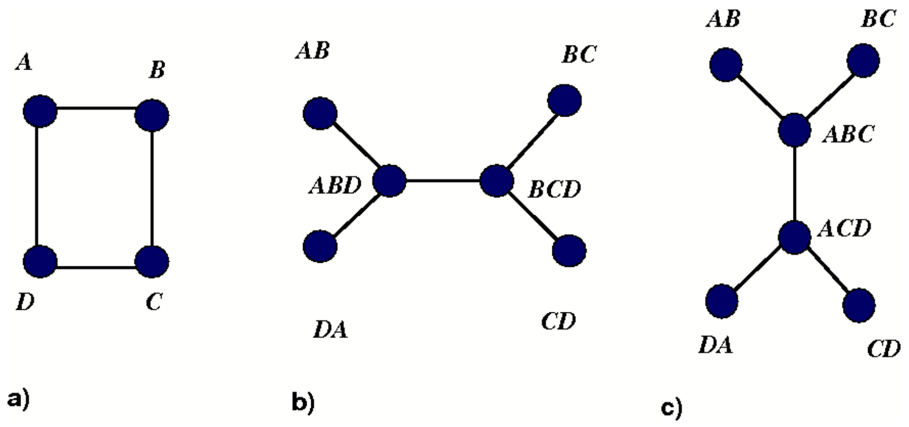


Fig. 1.5 The two possible (up to symmetry) topologies for an evolutionary tree for four taxa containing respectively characters: (A, B), (B, C), (C, D) and (D, A). In each case one pair of characters has to change state twice and the selection of such pair determines the topology of the tree.

are two possible topologies for the evolutionary tree for this set of taxa as illustrated in Figures 1.5(b)-(c). The number of character changes implied by each topology is the same. However, in the first case characters, B and D, have to change their state twice (and at least three of these character changes have to be deletions) while in the second case characters C and A have to change their state twice. If we knew which pair is more preserved in a given lineage relative to the other pair, we would be able to select the more likely topology. Similar situation occurs when we consider a larger cycle. This motivates the following concept of stability.

We say that a character is *stable* if it does not belong to a chordless cycle in the character overlap graph. Otherwise we say that the stability of the character is *challenged* and *number of challenges* is equal to the number of chordless cycles to which the character belongs. Note that the stability criterion can also identify characters which are preferentially conserved in one lineage but lost in many other lineages as stability of such characters is likely to be challenged by other characters.

Directly from the property of stability, we observe that the set of characters is stable only if the corresponding character overlap graph is chordal. In particular, it can be easily shown that a set of characters such that each character is gained at most once and lost at most once (called in [56] *persistent characters*) is stable [56]. Note that even the persistency criterion is significantly less stringent than the compatibility criterion discussed before as it allows for two changes of a character state.

Unfortunately, the problem of finding the minimum number of nodes whose removal leaves a graph chordal is NP complete [47]. To go around this problem, [56] use a simple heuristic. Namely, rather than considering all chordless cycles, they considered only squares. The squares were then eliminated by a greedy algorithm that iteratively removed characters belonging to the largest number of squares. After all squares are removed, they applied the Dollo parsimony (the maximum parsimony model that does not allow for multiple insertions of the same character) to construct the evolutionary tree based on the remaining characters.

The utility of a variant of this approach has been demonstrated by using it to construct the evolutionary tree from intron data compiled by Rogozin *et al.* [58]. This data contains information about introns found in conserved (and orthologous) genes of eight fully sequenced organisms: *Arabidopsis thaliana* (At), *Homo sapiens* (Hs), *C.elegans* (Ce), *Drosophila melanogaster* (Dm), *Anopheles gambaie* (Ag), *Saccharomyces cerevisiae* (Sc), *Schizosaccharomyces pombe* (Sp), and *Plasmodium falciparum* (Pf). Introns are identified by their starting position with respect to the coding sequence. The data contains 7236 introns; however most of these introns are observed in one organism only and thus are not informative. After eliminating these single-organism entries, 1790 introns were left. Define *intron pattern* to be a 0/1 vector of length eight which defines, for a given intron, which species have that intron and which do not. Note that with eight species there are $2^8 - 9$ different intron patterns (the subtraction corresponds to the assumption that each intron of interest must be in at least two species). Thus, some patterns are represented multiple times. The patterns that appear significantly more often than is expected by chance are considered to be more informative. Let n_i be the number of times pattern i is observed in the intron data, and r_i expected number of occurrences of the pattern by chance. Define $p_i = \frac{n_i}{r_i}$ to be the significance of the intron pattern i. Let S_i be the number of squares in which an intron with pattern i is involved. In this setting, the greedy square removal algorithm was set to remove iteratively intron patterns that maximize the value $\frac{S_i}{p_i}$. This provides a trade off between maximizing the number of removed squares and minimizing the significance of the removed intron patterns. The resulting evolutionary tree was consistent with the Coelomata hypothesis ([1, 8, 70, 16]). In contrast, the compatibility criterion failed to produce a meaningful tree in this case. The counterpart to the Coleometa hypothesis is the Ectysozoa hypothesis ([3, 48, 55, 34, 60]) (see Figure 1.6).

1.4 FORMATION OF MULTI-PROTEIN COMPLEXES

The complexity in biological systems arises not only from various individual protein molecules but also from their organization into systems with numerous interacting partners. In fact, most cellular processes are carried out by *multi-protein complexes*, groups of proteins that bind together to perform a specific task. Some proteins form stable complexes, such as the ribosomal complex that consists of more than 80 proteins and four RNA molecules, while other proteins form transient associations and are part of several complexes at different stages of a cellular process. A better understanding of this higher-order organization of proteins into overlapping complexes is an important step towards unveiling functional and evolutionary mechanisms behind biological networks.

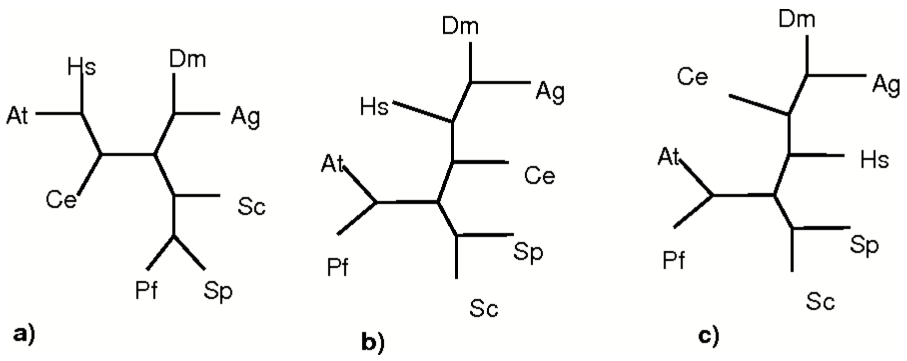


Fig. 1.6 Three tree topologies for organisms: *Arabidopsis thaliana* (At), *Homo sapiens* (Hs), *C.elegans* (Ce), *Drosophila melanogaster* (Dm), *Anopheles gambaie* (Ag), *Saccharomyces cerevisiae* (Sc), *Schizosaccharomyces pombe* (Sp), and *Plasmodium falciparum* (Pf) a) The incorrect Dollo parsimony tree computed from intron data b) The tree consistent with Coelomata hypothesis. This is also exactly the tree obtained after applying the squares removal procedure. c) The tree consistent with Ecdysozoa hypothesis.

Data on protein interactions are collected from the study of individual systems, and more recently through high-throughput experiments. There are many types of protein interactions, but in our quest to understand the dynamics of multi-protein complex formation we are mostly interested in physical protein interactions and interactions through being a member of the same protein complex, which we briefly review here.

There is a physical interaction between a pair of proteins if they come into a close contact or bind each other. High-throughput discovery of physical protein interactions is based on an experimental technique called *yeast two hybrid* (Y2H) [24]. To determine whether a pair of proteins, A and B, are able to physically interact, A is fused to a DNA binding domain and B is fused to a transcription activation domain. Physical interaction between A and B brings the DNA-binding domain and the transcription activation domain in proximity, which activates the transcription of the corresponding gene called a *reporter gene*. The expression level of the reporter gene is monitored and serves as a measure of physical interaction between proteins A and B. This technique was applied on a genome-wide scale to map physical protein interaction maps for several model organisms, most notably *Saccharomyces cerevisiae* [42, 67].

A pair of proteins may not physically interact but may still be members of the same protein complex. High-throughput discovery of this type of protein interaction is based on an experimental technique called *tandem affinity purification followed by mass spectrometry* (TAP/MS) [57]. In the TAP/MS approach, a protein of interest, which is called a *bait*, is tagged and used as a “hook” to pull out proteins that form a complex with it. These proteins are then identified by mass spectrometry techniques. The TAP/MS approach was used not only to map the interactome of *Saccharomyces* *cerevisiae* [40, 30, 29, 46], but also to study protein complexes involved in different signaling pathways [12].

Protein interactions are routinely represented by a graph, a *protein interaction network*, with vertices being the proteins and edges being the interactions. These graphs offer a static view of protein interactions in the cell, even though some proteins change their interacting partners and participate in different protein complexes. Can the topology of inherently static protein interaction network be used to elucidate the temporal order of dynamic multi-protein complex formation? In this section we review two such attempts: Farach-Colton *et al.* [22] used interval graphs to study the way in which various proteins join the ribosome maturation pathway, Zotenko *et al.* [72] used chordal graph and cographs to study the order in which various complexes are formed during cell signaling and other cellular processes.

1.4.1 Ribosomal Assembly

Ribosomes are massive molecular machines that are the major players in protein synthesis, they use a messenger RNA template to produce a polypeptide chain of newly created protein molecule. In eukaryotic cells ribosomes consists of two subunits, the so called 40S (small) and 60S (large) particles, which together account for four ribosomal RNAs and around 80 ribosomal proteins. Recent proteomic studies in *Saccharomyces cerevisiae* have identified around 200 auxiliary proteins that are involved in the assembly of ribosomal subunits but are not part of mature ribosomes. The ribosome synthesis is believed to proceed in an orderly pathway, the *ribosome assembly pathway*, and even though the main players of the pathway are known, little is known about the order in which these proteins join the pathway. For a mini-review see [21].

Farach-Colton and colleagues [22] proposed an interval model to represent the assembly pathway of the 60S ribosomal particle. In this model an auxiliary protein "enters" the pathway at some point and "leaves" the pathway at a latter point to never enter the pathway again. The model further assumes that a protein participates in the pathway through binding to other proteins currently in the pathway, therefore the assembly line can be thought of as an evolution of one protein complex to which proteins bind as they enter the pathway and from which proteins dissociate as they leave the pathway. Under this model the protein interaction network that spans the auxiliary proteins involved in the pathway should be an interval graph: each auxiliary protein is an interval and two proteins interact if and only if their interval overlap. Therefore the protein interaction network can be used to reconstruct the order in which the auxiliary proteins join the pathway.

Unfortunately, even if the proposed model captures correctly the ribosome assembly mechanism, experimental errors and incompleteness of protein interaction data may make the protein interaction network loose its interval graph property. To overcome this problem the authors use a variant of the multi-sweep LexBFS algorithm [18] to produce an ordering of vertices in the protein interaction network. The algorithm uses several iterations/sweeps of the LexBFS algorithm, where the first LexBFS sweep starts from an arbitrary vertex of the graph and every subsequent

LexBFS sweep uses the orderings produced by the previous iterations to choose the start vertex and break ties. If the network is an interval graph then the ordering produced by the algorithm is an I-ordering. If, on the other hand, the network is not an interval graph then the ordering as a whole won't be an I-ordering but it will induce an I-ordering on the vertices of some interval subgraph of the network; which subgraph would be correctly ordered depends on the order in which the vertices of the network are encountered by the algorithm. Thus, the authors suggests that computing an I-ordering of vertices of the graph is a reasonable step towards reconstruction the order in which the auxiliary proteins join the pathway.

The authors tested their approach on the protein interaction network spanning 96 auxiliary proteins involved in the assembly of the 60S particle. As part of the interaction data comes from TAP/MS experiments, it captures only interaction between the 25 bait proteins and other auxiliary proteins in a 96×25 protein interaction matrix. The rows/columns of the matrix were randomly permuted and supplied as an input to the multi-sweep LexBFS algorithm. The experiment was performed 5000 times and the rank of each protein in each of the 5000 orderings were recorded. Even though the input graph is not an interval graph only two different orderings emerged, which are denoted by $\mathcal{O}_1$ and $\mathcal{O}_2$. If an ordering of vertices is close to an I-ordering then the absolute difference in rank between any pair of adjacent vertices can not be arbitrarily large. Therefore, the authors establish significance of the two discovered orderings by the average difference in rank over two sets of protein interactions: a set of protein interactions comprising the network and thus seen by the algorithm, and a set of protein interactions not seen by the algorithm. The authors found that for both seen and unseen interactions the average difference for the $\mathcal{O}_1$ and $\mathcal{O}_2$ is significantly lower than average differences obtained with: (i) orderings produced by randomly permuting the proteins; (ii) orderings computed by the algorithm on random graph having the same degree distribution as the original input graph.

1.4.2 Multi-protein Complex Formation During Cell Signalling

In order to adapt to their environment, cells have to detect and respond to a vast variety of external stimuli. The detection and translation of these stimuli to a specific cellular response is achieved through a mechanism called *signal transduction pathway* or *signaling pathway*. The general principles of signal propagation through a pathway are common to almost all signaling pathways. First, an extracellular stimulus, usually a chemical ligand, binds to a membrane bound receptor protein. The energy from this interaction changes the state of the receptor protein, thus activating it. The active receptor is able to pass the signal to the *effector system* that generates the cell's response. A variety of proteins, the so-called *signalling proteins*, carry information between the receptor protein and the effector system. *Protein kinases*, for example, are special enzymes that add a phosphate group to certain residues of certain proteins through a process called *phosphorylation*, thus activating or suppressing the protein's ability to interact with other proteins.

The pattern of protein interaction during cell signaling is an excellent example of transient protein interactions and dynamic complex formation. For example, consider a sequence of events in one of the best studied signaling pathways, the mating pheromone signaling pathway in *Saccharomyces cerevisiae* (for more information see a review by Bardwell [4]). There are two mating types of yeast cells. When a yeast cell is stimulated by a pheromone secreted by a cell of an opposite mating type, it undergoes a series of physiological changes in preparation for mating, which include significant changes in gene expression of about 200 genes, oriented growth towards the partner, and changes in the cell-cycle. Signal propagation through the pathway is achieved through interaction of some 20 proteins, a schematic representation of the pathway and description of corresponding protein interactions are given in Figure 1.7.

Research efforts required to obtain the amount of detailed knowledge about a signaling pathway as is currently available for the mating pheromone pathway are enormous. Can the readily available high-throughput experimental data on protein interactions be used to elucidate some information about the pathway, such as the order of complex formation during signal propagation? In a recent work Zotenko *et al.* [72] have proposed a graph-theoretic method, Complex Overlap Decomposition (COD), that tries to recover the order of protein complex formation from the topology of protein interaction network that spans the pathway components. (The pathway components can be obtained from literature. Alternatively, putative pathway components can be automatically extracted from genome-wide protein interaction networks by computational methods [65, 62].)

The main idea behind the COD method, which is depicted in Figure 1.8, is to provide a representation of the protein interaction network that is analogous to a clique tree representation for chordal graphs, but in which nodes are cographs (representing functional groups) rather than maximal cliques (representing protein complexes). A *functional group* is either a protein complex (maximal clique in the protein interaction network) or a set of alternative variants of such complex. Such a representation accounts for two phenomena clearly illustrated in the pheromone signaling pathway described above: (i) the dynamic complex formation does not always follow a linear pathway but rather has a tree structure, where various branches correspond to the activation of different response systems; (ii) there may be several variants of a protein complex, such as MAPK complex centered at the scaffold protein which may include either $KSS1$ or $FUS3$ proteins but not both; It shoud be noted that cographs and their modular decomposition were previously used by Gagneur *et al.* to expose the hierarchical organization of protein complexes [27].

If a set of functional groups in a network were known then each functional group could be turned into a clique through addition of missing edges and clique tree construction algorithm could be applied to the modified network. As the functional groups are not known in advance, the authors propose a heuristic for their automatic delineation, where a set of edges is added to the network so that the maximal cliques in the modified network correspond to putative functional groups.

The COD method's edge addition strategy and its biological motivation builds on a functional interpretation of weak siblings in the network. Recall that a pair of nodes in a graph are weak siblings if they are not adjacent to each other but are adjacent to exactly the same set of nodes. In terms of protein interaction networks, weak siblings are proteins which interact with the same set of proteins but do not interact with each

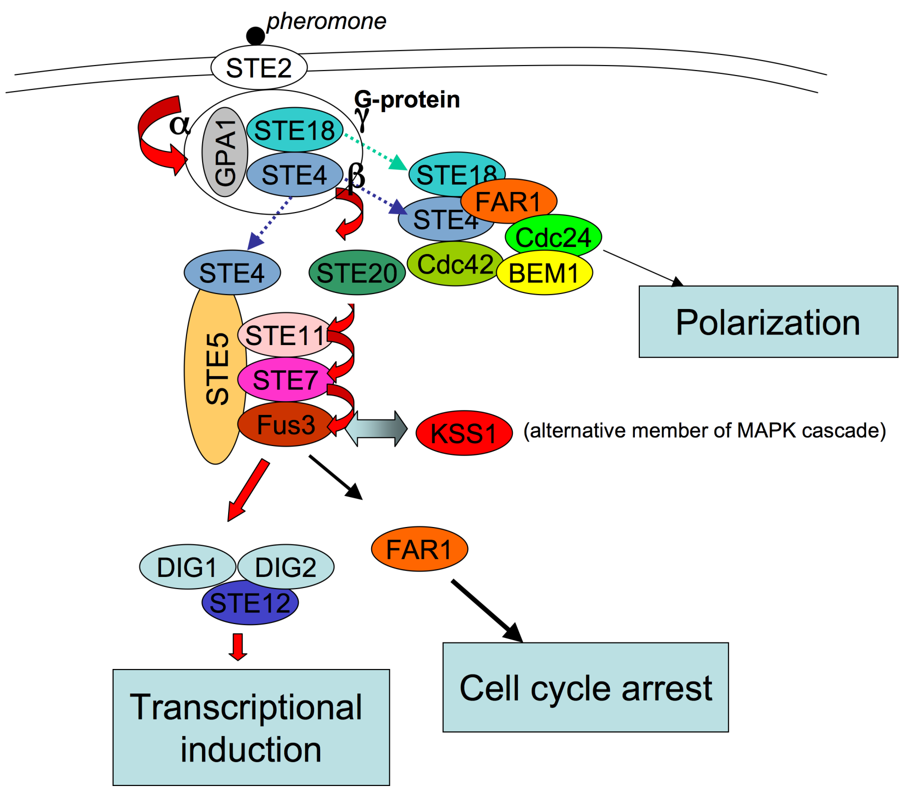


Fig. 1.7 A schematic representation of the key components of the pheromone signaling pathway assembled from information in [49, 37, 4]. A pheromone peptide binds a G-protein coupled receptor or GPCR (STE2/STE3). Activated receptor binds and activates a trimeric G-protein: G_α subunit (GPA1), G_β subunit (STE4) and G_γ subunit (STE18). The flow of information then proceeds via a three-tiered mitogen-activated protein kinase (MAPK) cascade and results in activation of STE12 transcription factor and subsequent upregulation of about 200 genes. The MAPK cascade also activates FAR1 protein, which is hypothesized to trigger a G_1 cell-cycle arrest through an interaction with CDC28, a master regulator of the cell-cycle. The MAPK cascade consists of three protein kinases STE11, STE7 and either FUS3 or KSS1, which activate each other sequentially through phosphorylation. Thus STE11 activates STE7, which in turn activates either FUS3 or KSS1. The phosphorylation process is enhanced through a presence of a scaffold protein STE5, which binds and thus co-localizes all three components of the MAPK cascade. Activated FUS3 and KSS1 proteins in turn bind their substrates, DIG1/DIG2/STE12 complex and FAR1 protein. Another branch of the pathway, which includes proteins STE4, STE18, FAR1, CDC24, CDC42, and BEM1 is responsible for triggering a "polarized growth towards the mating partner" or polarization response.

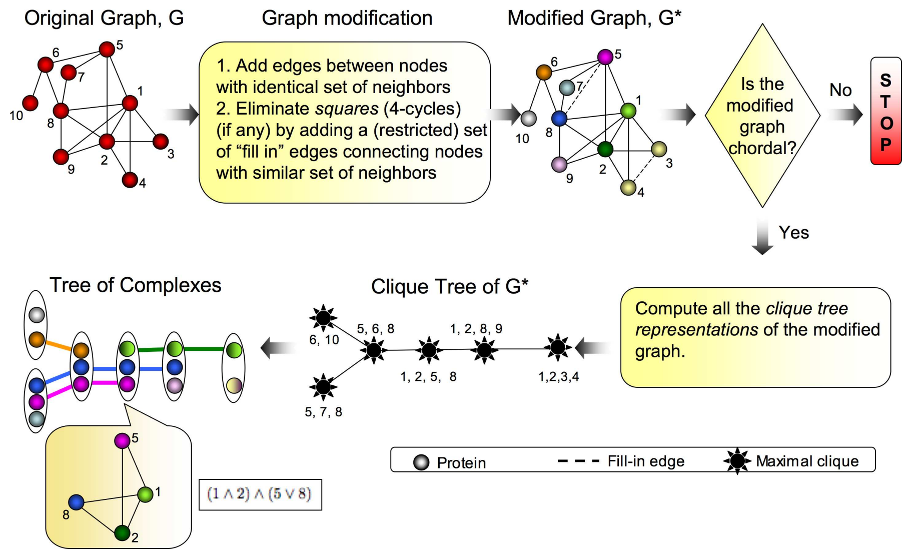

Original Graph, G
Graph modification
1. Add edges between nodes with identical set of neighbors
2. Eliminate *squares* (4-cycles) (if any) by adding a (restricted) set of "fill in" edges connecting nodes with similar set of neighbors
Modified Graph, G*
Is the modified graph chordal?
No
STOP
Yes
Compute all the *clique tree representations* of the modified graph.
Clique Tree of G*
6, 10
5, 6, 8
1, 2, 8, 9
1, 2, 5, 8
1,2,3,4
5, 7, 8
Tree of Complexes
(1 ∧ 2) ∧ (5 ∨ 8)
Protein
Fill-in edge
Maximal clique

Fig. 1.8 An illustration of the Complex Overlap Decomposition (COD) method. An edge, $(3, 4)$, connecting a pair of weak siblings is added to the graph. A fill-in edge between proteins 5 and 8 is added to eliminate all five 4-cycles in the graph: $\{5, 6, 8, 7\}$, $\{1, 5, 7, 8\}$, $\{2, 5, 7, 8\}$, $\{1, 5, 6, 8\}$, and $\{2, 5, 6, 8\}$. If the modified graph is chordal, all clique tree representations are computed and each such representation is extended into a *Tree of Complexes* representation of the original graph. The Tree of Complexes is constructed by projecting each maximal clique in the modified graph, G^*, to a functional group in the original graph G. For example, a four node maximal clique, $\{1, 2, 5, 8\}$, in G^* is projected to a four node functional group in G, by removing a fill-in edge $(5, 8)$. Each functional group is represented by a Boolean expression, such as $(1 \wedge 2) \wedge (5 \vee 8)$, which means that the functional group contains two variants of a complex, $\{1, 2, 5\}$ and $\{1, 2, 8\}$. This figure is reproduced from [72].

other. In particular, proteins that can substitute for each other in a protein interaction network may have this property. Similarly, weak siblings may correspond to a pair of proteins that belong to the same protein complex but are not connected by an edge due to missing data or an experimental error. Therefore, the heuristic first connects every pair of weak siblings by an edge. If the modified graph is not chordal an additional set of edges that connect pairs of proteins close to being weak siblings is added; each such edge is a diagonal in one or more *squares*, chordless cycles of length four, in the graph. The heuristic finds a minimum cost set of diagonals that eliminates all the squares in the graph, where the cost of a diagonal is inversely proportional to the amount of overlap between the neighborhoods of its endpoints.

If the modification step succeeds, i.e., the modified graph is chordal, all the clique tree representations of the modified graph are constructed and then extended to the *Tree of Complexes representations* of the original graph. The COD algorithm keeps track of all the edge additions and uses this information to delineate functional groups by projecting each maximal clique onto the original network and removing all introduced edges contained in the clique. For example, in the modified graph of Figure 1.8 a maximal clique with four nodes, $\{1, 2, 5, 8\}$, is projected to a functional group by removing an edge connecting protein 5 and 8. This functional group contains two variants of a protein complex, $\{1, 2, 5\}$ and $\{1, 2, 8\}$, which are compactly represented by the Boolean expression $(1 \wedge 2) \wedge (5 \vee 8)$. If, on the other hand, the modified graph is not chordal, the COD method stops without producing the representation.

The authors demonstrated the effectiveness of their approach by decomposing protein interaction networks for two signaling pathways: the mating pheromone signaling pathway and the NF-kB signaling pathway. Here we apply the COD method to the pheromone signaling pathway, where the pathway components were taken from [4] (Table 1) and protein interactions that span the pathway components from the DIP database [61] (version 01/16/2006; core set of interactions). The network is shown in Figure 1.9(a). Since proteins STE2/STE3 are disconnected from the rest of the components, we have removed them from the network in our analysis. The COD method adds three diagonals to eliminate eleven squares in the network: (STE4,BEM1), (FUS3, KSS1), and (GPA1, STE5), which results in twelve functional groups listed in Figure 1.9 along with the corresponding Boolean expressions. There are twelve Tree of Complexes representations for this protein interaction network one of which is shown in Figure 1.9(b). All the representations agree on the interconnection pattern between functional groups, $B-E$, H, and $J-L$. The difference between various tree variants comes from how functional groups A, $F-G$, and I are connected to the rest of the tree: (i) functional group A can be attached either through (A, C), or (A, B), or (A, J); (ii) functional group I through (I, E), or (I, D); (iii) functional group F through (F, E) or (F, H).

Compare the representation in Figure 1.9(b) to the schematic representation of the pheromone signaling pathway shown in Figure 1.7. Using only protein interaction information the COD method was able to recover two branches of the pathway, the MAPK cascade branch and the polarization branch. The MAPK cascade branch

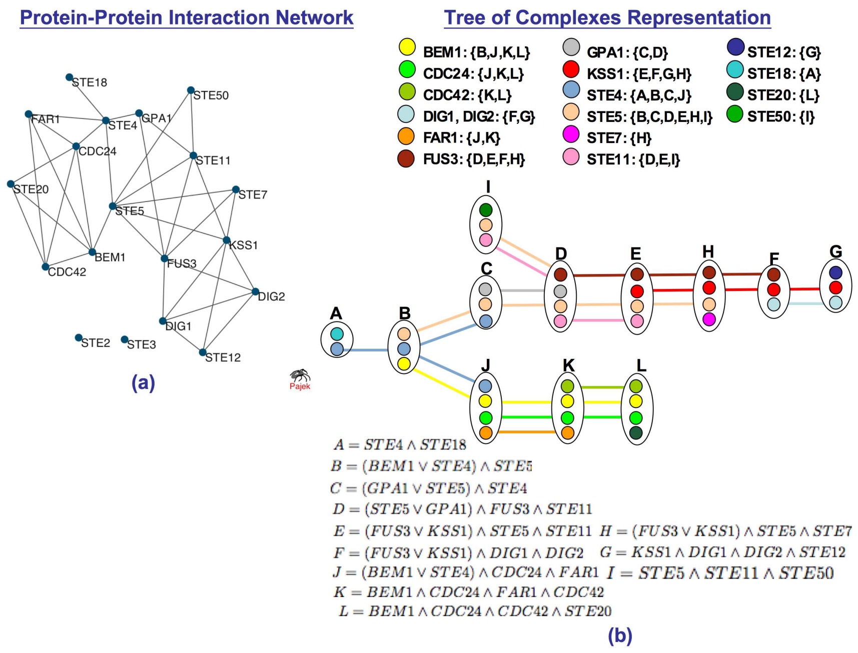

Protein-Protein Interaction Network
STE18
STE50
FAR1
STE4
GPA1
CDC24
STE11
STE20
STE7
STE5
KSS1
BEM1
FUS3
CDC42
DIG2
DIG1
STE2
STE3
STE12
(a)
Pajek
Tree of Complexes Representation
BEM1: {B,J,K,L}
CDC24: {J,K,L}
CDC42: {K,L}
DIG1, DIG2: {F,G}
FAR1: {J,K}
FUS3: {D,E,F,H}
GPA1: {C,D}
KSS1: {E,F,G,H}
STE4: {A,B,C,J}
STE5: {B,C,D,E,H,I}
STE7: {H}
STE11: {D,E,I}
STE12: {G}
STE18: {A}
STE20: {L}
STE50: {I}
A B C D E F G H I J K L
$A = STE4 \wedge STE18$
$B = (BEM1 \vee STE4) \wedge STE5$
$C = (GPA1 \vee STE5) \wedge STE4$
$D = (STE5 \vee GPA1) \wedge FUS3 \wedge STE11$
$E = (FUS3 \vee KSS1) \wedge STE5 \wedge STE11$
$H = (FUS3 \vee KSS1) \wedge STE5 \wedge STE7$
$F = (FUS3 \vee KSS1) \wedge DIG1 \wedge DIG2$
$G = KSS1 \wedge DIG1 \wedge DIG2 \wedge STE12$
$J = (BEM1 \vee STE4) \wedge CDC24 \wedge FAR1$
$I = STE5 \wedge STE11 \wedge STE50$
$K = BEM1 \wedge CDC24 \wedge FAR1 \wedge CDC42$
$L = BEM1 \wedge CDC24 \wedge CDC42 \wedge STE20$
(b)

Fig. 1.9 The mating pheromone signaling pathway. **(a)** The protein interaction network for the components of the pathway. The network was drawn with Pajek [5]. **(b)** One of the twelve possible Tree of Complexes representations for the network.The activation of the pathway corresponds to node A in the tree which contains the G_β (STE4) protein. From node A, the Tree of Complexes splits into two branches. One branch roughly corresponds to the MAPK cascade activated response, while another branch roughly corresponds to the polarization response. The MAPK cascade branch spans four nodes in the tree: I, D, E, and H. The activation of transcription factor complex by $FUS3$ and $KSS1$ is in nodes F and G. The polarization branch spans nodes J, K and L.

spans four nodes in the tree: I, D, E, and H. The polarization branch spans nodes J, K and L.

Acknowledgments

This work was supported by the Intramural Research Program of the NIH, National Library of Medicine.

References

1. A. Adoutte, G. Balavoine, N. Lartillot, O. Lespinet, B. Prud'homme, and R. de Rosa. The new animal phylogeny: reliability and implications. *Proceedings of the National Academy of Sciences U S A*, 97:4453–4456, 2000.
2. R. Agarwala and D. Fernandez-Baca. A polynomial time algorithm for the perfect phylogeny problem when the number of character states is fixed. *SIAM Journal on Computing*, 23:1216–1224, 1994.
3. A. M. Aguinaldo, J. M. Turbeville, L. S. Linford, M. C. Rivera, J. R. Garey, R. A. Raff, and J. A. Lake. Evidence for a clade of nematodes, arthropods and other moulting animals. *Nature*, 387:489–493, 1997.
4. L. Bardwell. A walk-through of the yeast mating pheromone response pathway. *Peptides*, 26:339–350, 2005.
5. V. Batagelj and A. Mrvar. Pajek - Program for large network analysis. *Connections*, 2:47–57, 1998.
6. S. Benzer. On the topology of genetic fine structure. *Proceedings of the National Academy of Sciences USA*, 45:1607–1620, 1959.
7. P.A. Bernstein and N. Goodman. Power of natural semijoins. *SIAM Journal of Computing*, 10:751–771, 1981.
8. J.E. Blair, K. Ikeo, T. Gojobori, and S.B. Hedges. The evolutionary position of nematodes. *BMC Evolutionary Biology*, 2:7, 2002.
9. H. Bodlaender, M. Fellows, and T.J. Warnow. Two strikes against perferct phylogeny. In *ICALP*, 1992.
10. H. Bodlaender and T. Kloks. A simple linear time algorithm for triangulating three-colored graphs. *Journal of Algorithms*, 15:160–172, 1993.
11. K.S. Booth and G.S. Lueker. Testing for consecutive ones property, interval graphs, and graph planarity using PQ-Tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976.
12. T. Bouwmeester, A. Bauch, H. Ruffner, P.O. Angrand, G. Bergamini, K. Croughton, C. Cruciat, D. Eberhard, J. Gagneur, and S. Ghidelli. A physical

and functional map of the human TNF-alpha/NF-kappaB signal transduction pathway. *Nature Cell Biology*, 6:97–105, 2004.

13. P. Buneman. A characterization of rigid circuit graphs. *Discrete Mathematics*, 9:202–215, 1974.

14. L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58:171–176, 1996.

15. L. L. Cavalli-Sforza and A.W.F. Edwards. Phylogenetic analysis. Models and Estimation Procedures. *Evolution*, 21:550–570, 1967.

16. F. D. Ciccarelli, T. Doerks, C. von Mering, C.J. Creevey, B. Snel, and P. Bork. Toward automatic reconstruction of a highly resolved tree of life. *Science*, 311:1283–1287, 2006.

17. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT Press, second edition, 2001.

18. D.G. Corneil, S. Olariu, and L. Stewart. The ultimate interval graph recognition algorithm? In *Proceedings of the 9th ACM-SIAM Simposium on Discrete Algorithms (SODA)*, 1998.

19. D.G. Corneil, Y. Perl, and L. Stewart. Complement reducible graphs. *Discrete Applied Mathematics*, 3:163–174, 1981.

20. D.G. Corneil, Y. Perl, and L.K. Stewart. A linear time recognition algorithm for cographs. *SIAM Journal on Computing*, 14:926–934, 1985.

21. M. Dlakic. The ribosomal subunit assembly line. *Genome Biology*, 6:234, 2005.

22. M. Farach-Colton, Y. Huang, and J.L.L. Woolford. Discovering temporal relations in molecular pathways using protein-protein interactions. In *Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology*, pages 150–156, San Diego, California, USA, 2004.

23. J. Felsenstein. *Inferring phylogenies*. Sinauer Associates, 2004.

24. S. Fields and O. Song. A novel genetic system to detect protein-protein interactions. *Nature*, 340:245–246, 1989.

25. W.M. Fitch. Toward defining the course of evolution: Minimum change for a specified tree topology. *Systematic Zoology*, 20:406–416, 1971.

26. D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15:835–855, 1965.

27. J. Gagneur, R. Krause, T. Bouwmeester, and G. Casari. Modular decomposition of protein interaction networks. *Genome Biology*, 5:R57, 2004.

28. M.R. Garey and D.S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Company, 1979.

29. A.C. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L.J. Jensen, S. Bastuck, B. Dümpelfeld, A. Edelmann, M.A. Heurtier, V. Hoffman, C. Hoefert, K. Klein, M. Hudak, A.M. Michon, M. Schelder, M. Schirle, M. Remor, T. Rudi, S. Hooper, A. Bauer, T. Bouwmeester, G. Casari, G. Drewes, G. Neubauer, J.M. Rick, B. Kuster, P. Bork, R.B. Russell, and G. Superti-Furga. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440:631–636, 2006.

30. A.C. Gavin, M. Bosche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, J.M. Rick, A.M. Michon, and C.M. Cruciat. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–147, 2002.

31. F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory (B)*, 16:47–56, 1974.

32. F. Gavril. Generating the maximum spanning trees of a weighted graph. *Journal of Algorithms*, 8:592–597, 1987.

33. A. George, J.R. Gilbert, and Liu J.W.H., editors. *Graph theory and sparse matrix computations*, chapter 1. Springer, New York, 1993.

34. G. Giribet, D. L. Distel, M. Polz, W. Sterrer, and W. C. Wheeler. Triploblastic relationships with emphasis on the acoelomates and the position of Gnathostomulida, Cycliophora, Plathelminthes, and Chaetognatha: A combined approach of 18S rDNA sequences and morphology. *Systematic Biology*, 49(3):539–562, Sep 2000.

35. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier, second edition, 2004.

36. M.C. Golumbic and I.B-A.Hartman, editors. *Graph theory, combinatorics and algorithms: interdisciplinary applications*, chapter 4, pages 63–105. Springer, 2005.

37. A. Gruhler, J.V. Olsen, S. Mohammed, P. Mortensen, N.J. Faergeman, M. Mann, and O.N. Jensen. Quantitative phosphoproteomics applied to the yeast pheromone signaling pathway. *Molecular and Cellular Proteomics*, 4(3):310–327, Mar 2005.

38. D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19–28, 1991.

39. C. Ho and R.C.T. Lee. Counting clique trees and computing perfect elimination schemes in parallel. *Information Processing Letters*, 31:61–68, 1989.

40. Y. Ho, A. Gruhler, A. Heilbut, G.D. Bader, L. Moore, S.L. Adams, A. Millar, P. Taylor, K. Bennett, and K. Boutilier. Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry. *Nature*, 415:180–183, 2002.

41. R. Idury and A. Schaffer. Triangulating three-colored graphs in linear time and linear space. *SIAM Journal on Discrete Mathematics*, 6:289–294, 1993.

42. T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences USA*, 98:4569–4574, 2001.

43. S. Kannan and T.J. Warnow. Triangulating three-colored graphs. *SIAM Journal on Discrete Mathematics*, 5:249–258, 1992.

44. S. Kannan and T.J. Warnow. Inferring evolutionary history from DNA sequences. *SIAM Journal of Computing*, 23:713–737, 1994.

45. H. Kaplan, R. Shamir, and R.E.Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal on Computing*, 28:1906–1922, 1999.

46. N.J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A.P. Tikuisis, T. Punna, J.M. Peregrín-Alvarez, M. Shales, X. Zhang, M. Davey, M.D. Robinson, A. Paccanaro, J.E. Bray, A. Sheung, B. Beattie, D.P. Richards, V. Canadien, A. Lalev, F. Mena, P. Wong, A. Starostine, M.M. Canete, J. Vlasblom, S. Wu, C. Orsi, S.R. Collins, S. Chandran, R. Haw, J.J. Rilstone, K. Gandi, N.J. Thompson, G. Musso, P. St Onge, S. Ghanny, M.H.Y. Lam, G. Butland, A.M. Altaf-Ul, S. Kanaya, A. Shilatifard, E. O'Shea, J.S. Weissman, C. J. Ingles, T.R. Hughes, J. Parkinson, M. Gerstein, S.J. Wodak, A. Emili, and J.F. Greenblatt. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature*, 440(7084):637–643, Mar 2006.

47. J.M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.*, 20:219–230, 1980.

48. J. Mallatt and C.J. Winchell. Testing the new animal phylogeny: first use of combined large-subunit and small-subunit rRNA gene sequences to classify the protostomes. *Molecular Biology and Evolution*, 19:289–301, 2002.

49. D. Matheos, M. Metodiev, E. Muller, D. Stone, and M.D. Rose. Pheromone-induced polarization is dependent on the Fus3p MAPK acting through the formin Bni1p. *Journal of Cell Biology*, 165(1):99–109, Apr 2004.

50. T. A. McKee and F.R. McMorris. *Topics in intersection graph theory*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, 1999.

51. F.R. McMorris, T.J. Warnow, and T. Wimer. Triangulating vertex-colored graphs. *SIAM Journal on Discrete Mathematics*, 7:196–306, 1994.

52. M. Middendorf, E. Ziv, and C.H. Wiggins. Inferring network mechanisms: The Drosophila melanogaster protein interaction network. *Proceedings of the National Academy of Sciences USA*, 102:3192–3197, 2005.

53. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

54. R.H. Mohring and F.J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Annals of Discrete Mathematics*, 19:257–356, 1984.

55. K. J. Peterson and D. J. Eernisse. Animal phylogeny and the ancestry of bilaterians: Inferences from morphology and 18S rDNA gene sequences. *Evolution and Development*, 3(3):170–205, 2001.

56. T.M. Przytycka. An important connection between network motifs and parsimony models. In *RECOMB*, 2006.

57. G. Rigaut, A. Shevchenko, B. Rutz, M. Wilm, M. Mann, and B. Séraphin. A generic protein purification method for protein complex characterization and proteome exploration. *Nature Biotechnology*, 17(10):1030–1032, Oct 1999.

58. I.B. Rogozin, Y.I. Wolf, A. V. Sorokin, B.G. Mirkin, and E.V. Koonin. Remarkable interkingdom conservation of intron positions and massive, lineage-specific intron loss and gain in eukaryotic evolution. *Current Biology*, 13:1512–1517, 2003.

59. D.J. Rose and R.E. Tarjan. Algorithmic aspects of vertex elimination. *SIAM Journal of Applied Mathematics*, 34:176–197, 1978.

60. S.W. Roy and W. Gilbert. Resolution of a deep animal divergence by the pattern of intron conservation. *Proceedings of the National Academy of Sciences USA*, 102(12):4403–4408, Mar 2005.

61. L. Salwinski, C.S. Miller, Smith A.J., F.K. Pettit, J.U. Bowie, and Eisenberg D. The Database of Interacting Proteins: the 2004 update. *Nucleic Acids Research*, 32:D449–D451, 2004.

62. J. Scott, T. Ideker, R.M. Karp, and R. Sharan. Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology*, 13(2):133–144, Mar 2006.

63. R. Shamir. Advanced topics in graph theory. Technical report, Tel-Aviv University, 1994.

64. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31, 2002.

65. M. Steffen, A. Petti, J. Aach, P. D'haeseleer, and G. Church. Automated modelling of signal transduction networks. *BMC Bioinformatics*, 3:34, 2002.

66. R.E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing*, 13(3):566–579, 1984.

67. P. Uetz, L. Giot, G. Cagney, T.A. Mansfield, R.S. Judson, J.R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, and P. Pochart. A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature*, 403:623–627, 2000.

68. J.R. Walter. *Representation of rigid circuit graphs*. PhD thesis, Wayne State University, 1972.

69. E. O. Wilson. A consistency test for phylogenies based on contemporaneous species. *Systematic Zoology*, 14:214–220, 1965.

70. Y.I. Wolf, I.B. Rogozin, and E.V. Koonin. Coelomata and not Ecdysozoa: Evidence from genome-wide phylogenetic analysis. *Genome Research*, 14:29–36, 2004.

71. M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 2:77–79, 1981.

72. E. Zotenko, K.S. Guimarães, R. Jothi, and T.M. Przytycka. Decomposition of overlapping protein complexes: A graph theoretical method for analyzing static and dynamic protein associations. *Algorithms for Molecular Biology*, 1(1):7, 2006.